Tips en tricks voor

Voor de serie “Pre-Emptive State Machine”

**Noot:** kan gedownload worden van:

https://github.com/knijff1961/MVA

Auteurs:

Datum : 04-08-2023

Versie : 0.9

Status : Review

Inhoud

[Content 2](#_Toc142038671)

[Afkortingen 3](#_Toc142038672)

[1 Introductie 4](#_Toc142038673)

Afkortingen

|  |  |
| --- | --- |
| MVA | Modelbouwvereniging Arnhem en omstreken |
| IDE | Integrated Development Environment; IDE is software die een ontwikkelaar helpt bij het ontwikkelen van software |
| Sketch | De naam die Arduino gebruikt voor een programma. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introductie

Dit document beschrijft allerlei tips en tricks om software voor de Arduino beter te kunnen ontwikkelen. Dit document is geschreven voor de artikelen reeks “Pre-Emptive State Machine” welke in het kwartaalblad van de MVA zal verschijnen. In deze reeks wordt beschreven hoe een Arduino meerdere taken (schijnbaar) tegelijkertijd kan uitvoeren.

Dit document gaat ervan uit dat de sketches ontwikkeld worden op een MS Windows omgeving. Voor andere systemen zal zelf gezocht moeten worden op internet.

Dit document is vooral tot stand gekomen omdat verschillende vragen naar boven gekomen zijn welke ook gesteld kunnen worden door anderen. In dit document zal dan dieper ingegaan worden op deze vragen.

Dit document zal dus regelmatig geüpdatet worden.

# De Arduino IDE

## Bestand locaties

Dit hoofdstuk beantwoord de vragen waar bepaalde bestanden binnen de Arduino IDE te vinden zijn of gemaakt moeten worden.

### Waar vind ik de libraries map

Deze map kan op drie plaatsen gevonden worden:

* In de installatie map waar de IDE geïnstalleerd is.
* De Arduino map in Documenten
* In de directory waar de sketch gemaakt is

#### Vinden van de installatie map

De installatie map is meestal “C:\Program Files (x86)\Arduino\libraries” tenzij deze op een andere plaats geïnstalleerd is. (Tijdens installatie is een andere map gekozen.) Om deze toch te kunnen vinden kunnen de volgende stappen gedaan worden:

|  |  |
| --- | --- |
| In de taakbalk zoek voor Arduino: |  |
| In de lijst van Beste overeenkomst is de Arduino App te vinden. |  |
| Klik met de rechter muisknop op deze Arduino App en selecteer “Bestandslocatie openen”. De bestandslocatie van het start-menu wordt geopend. |  |
| Klik met de rechtermuis op Arduino |  |
| Selecteer “Eigenschappen” |  |
| Nu zal er een venster geopend worden. Hierin is de locatie van de Arduino te vinden (Doel).  Ga naar deze locatie. |  |

De locatie in dit geval is (natuurlijk) “C:\Program Files (x86)\Arduino”. Hierin is de map “libraries” te vinden. De library map is dus: “C:\Program Files (x86)\Arduino\libraries”

#### De Arduino map in Documenten

Standaard is deze map: ~/Documenten/Arduino/libraries/

(in het Engels: ~/Documents/Arduino/libraries/)

#### In de directory waar de sketch gemaakt is

Wanneer een nieuwe sketch gemaakt wordt dan zal deze in een map geplaatst worden. Binnen deze map mag een libraries map gemaakt worden waarin de verschillende libraries geplaatst kunnen worden.

**Dit wordt echter afgeraden**

De reden is dat voor elke sketch een nieuwe libraries map gemaakt moet worden waarin de libraries geplaatst moeten worden. Wanneer later een verbeterde library gebruikt zal worden, dan zal dit in ALLE libraries mappen doorgevoerd moeten worden. Dit is erg omslachtig.

### Waar kan ik de sketch map vinden?

#### De standaard sketch map

In de Arduino IDE selecteer de menu Bestand. Hierin klik op voorkeuren:

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijving

De standaard Sketchbooklocatie kan nu gelezen worden:

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

#### Waar kan ik mijn huidige sketch vinden

Selecteer de sketch van welke de bestandslocatie gevonden moet worden.

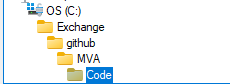
In de menu Bestand selecteer opslaan als.



Een venster wordt geopend. Klik op de drop-down lijst “Opslaan in”



Onderaan is dan de map te vinden:



Noot: natuurlijk is de locatie voor elke sketch verschillend. Dit is alleen maar een voorbeeld.

## (Her)installatie van de libraries

In dit hoofdstuk moet de library map gevonden worden waarin de bijgeleverde libraries geïnstalleerd zijn of moeten worden. Zie hiervoor Waar vindt ik de libraries map

Wanneer de libraries nog niet geïnstalleerd zijn dan kan direct naar De installatie van de libraries gegaan worden.

### Verwijderen van de libraries

De meegeleverde libraries kunnen eenvoudig verwijderd worden door eerst naar de library map te gaan.

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijvingDe volgende drie libraries moeten verwijderd worden:

### De installatie van de libraries

Wanneer de files van GitHub gedownload zijn, dan staan in de directory “Libraries” drie mappen die hierboven getekend zijn. Kopieer deze mappen naar de library directory van de Arduino.

Dat is alles: de libraries kunnen nu gebruikt worden.

## Voorbeelden in de library

Wanneer een library geïnstalleerd wordt, dan zal bij deze libraries ook voorbeeld code meegeleverd worden. Deze voorbeelden kunnen eenvoudig geopend worden en als voorbeeld dienen om een eigen schets te bouwen.

Een voorbeeld code document is de debounce test van de CDebounce library. Wanneer de library goed geïnstalleerd is, dan kunnen de voorbeelden eenvoudig gevonden worden in het menu Bestand/Voorbeelden:

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Vervolgens kan in de rechter sub-menu de CDebounce library gevonden worden. (In de sectie ‘Voorbeelden van Custom Libraries’):

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

De DebounceDetect is nu eenvoudig te openen. Let erop: eventueel aangepaste code kan door de Arduino IDE niet veranderd worden; deze code is een voorbeeld maar kan wel via kopiëren en plakken naar een nieuwe schets over gekopieerd worden. In de nieuwe schets is de code dan wel te veranderen.

# Pre-emptive code

## De basis

# Genereren van classes in de library

Wanneer een generieke statemachine ontwikkeld is, bijvoorbeeld een knipperinstallatie of een knop, dan kan deze in meerdere schetsen gebruikt worden. Het is niet handig om elke keer in een nieuwe schets de code te kopiëren.

Hiervoor kan beter een library class gemaakt worden.

Om een library class te kunnen maken zijn verschillende stappen mogelijk. Hier wordt een mogelijke stappenplan gemaakt. Als basis wordt de zgn. debounce probleem gebruikt. Wat debounce precies is wordt uitgelegd in: CDebounce. Hierin wordt ook de statemachine en de Frizing schema beschreven.

## Preemptive loop code

Code is geschreven in: ~\Code\Adds\Debounce\DebounceTest\_v2.ino

Eerst wordt begonnen met het programmeren van preemptive code in de loop functie. De statemachine voor de debounce staat in Het oplossen van bouncing (debouncing); deze statemachine is geïmplementeerd in de file.

Duidelijk zijn de 4 states te onderscheiden; alleen case 1 & case 3 hebben een init en wait sub-state. De code is redelijk recht toe recht aan; wanneer de button een andere waarde aangeeft dan moet deze een tijdlang (bijvoorbeeld 100ms) dezelfde waarde aanhouden voordat de code aangeeft dat de status veranderd is. Na deze tijd kan men aannemen dat de knop aan c.q. uit is.

In de code is de tijd van debounce detectrie gezet op 1 seconde:

int iDebounceTime = 1000;

Dit is een erg lange tijd; normaal is dit ongeveer 100ms echter, om te kunnen controleren of de code werkt kan 1000ms gebruikt worden totdat de code correct is.

De volgende stappen moeten gedaan worden (en voor alle overige sub secties:

* Maak de schakeling door gebruik te maken van de Frizling schema
* Upload de schets in de Arduino
* Druk de knop in, langer dan één seconde: de led gaat aan
* Laat de knop los, ook langer dan één seconde: de led gaat uit
* Druk de knop snel achter elkaar in (binnen één seconde): de led blijft uit.
* Druk de knop in, langer dan één seconde: de led gaat aan
* Terwijl de led aan is, laat binnen één seconde meermalen de knop los en wer ingedrukt. De led blijft aan.
* Experimenteer met meerder keren de knop in te drukken en weer los te laten.

Wat op moet vallen is dat de led aangat wanneer langer dan één seconde de knop ingedrukt is en weer uitgaat nadat de knop langer dan één seconde is losgelaten.

## Omzetten van loop code naar een functie

Code is geschreven in: ~\Code\Adds\Debounce\DebounceTest\_v3.ino

De tweede stap is het omschrijven van de loop code naar een functie. In dit geval is dat vrij eenvoudig:

Maak een functie aan en noem deze go met een argument genaamd currentMillis; de declaratie is dus:

void go(unsigned long currentMillis) {

…

}

En in de loop staat dan een aanroep naar deze functie:

void loop() {

unsigned long currentMillis = millis();

**go(currentMillis);**

if(bState)

digitalWrite(PIN\_LED\_DEBOUNCED,HIGH);

else

digitalWrite(PIN\_LED\_DEBOUNCED,LOW);

}

De overige code is om de status weer te geven:

bSstate=false wanneer led uit moet zijn (knop langer dan één seconde niet ingedrukt)

bSstate=true wanneer led aan moet zijn (knop langer dan één seconde ingedrukt)

Upload de schets en doe de testen om er zeker van te zijn dat de functionaliteit hetzelfde is.

## Zet de code om naar een class

Code is geschreven in: ~\Code\Adds\Debounce\DebounceTest\_v4.ino

Op internet zijn vele, vele sites te vinden hoe een class in C++ gemaakt moet worden. Deze sub-sectie neemt aan dat de basis kennis van het maken van een class aanwezig is.

Web pagina’s voor het creëren van een class zijn o.a.:

De web site w3schools is een site die vele, vele programmeer talen eenvoudig uitlegt (wel in het Engels). Ook de basis van C++ wordt hier uitgelegd:

<https://www.w3schools.com/cpp/default.asp>

De site geeft de mogelijkheid om, naast het leren, ook te experimenteren met code. De link die specifiek de classes uitlegt is:

<https://www.w3schools.com/cpp/cpp_oop.asp>

Om een class voor de Arduino te maken (die later in de library gezet kan worden) moet eerst nagedacht worden WAT de class moet doen.

De volgende class requirements (eisen) zijn opgesteld:

* De class moet weten aan welke pin de button hangt; deze pin nummer moet dus een onderdeel zijn van de class. De pinnummer wordt tijdens het creëren van een class object meegegeven worden.
* De class moet weten welke time-out tijd gewacht moet worden. Deze time-out kan gegeven worden door:
  + Tijdens het creëren van het object.
  + Via een functie
* Ook kan de time-out tijd gevraagd worden aan het object.
* De class zal de (debounced) status van de knop moeten geven.

**Noot:** de PIN\_LED\_DEBOUNCED komt NIET in de clas voor dit omdat deze alleen gebruikt wordt voor test doeleinden; de debounce class moet ook voor andere zeken toe te passen zijn.

Deze gegevens worden dus gedeclareerd in de header (\*.h) sectie. De implementatie komt dan in de code sectie.

### De header sectie

De header sectie definieert wat een object voor gegevens kan verwerken. Het bevat dus:

* Hoe de class gedigitaliseerd wordt (constructors)
* De functies (methods) van de class
* De variabelen (properties) in de class

De naam van de class is logisch gekozen: CDebounce

De class definitie ziet er als volgt uit:

class CDebounce {

public:

CDebounce(int iPinButton);

CDebounce(int iPinButton, int iDebounceTime);

void go(unsigned long currentMillis);

bool getState();

int getDebounceTime();

void setDebounceTime(int iDebounceTime);

private:

CPreEmptiveTimer cPreEmptiveTimer;

int \_iPinButton;

int \_iDebounceTime = 100;

bool \_bState = false;

};

#### De constructors

Deze class heeft 2 constructors

##### CDebounce(int iPinButton)

Het object zal gecreëerd worden waarbij alleen de pin nummer doorgegeven wordt. De standaard debounce tijd zal in dit geval 100 zijn.

De standaard debounce tijd wordt vastgelegd bij de properties (zie int \_iDebounceTime = 100)

##### CDebounce(int iPinButton, int iDebounceTime);

Het object zal gecreëerd worden waarbij niet alleen de pin nummer doorgegeven wordt maar ook de debounce tijd.

De standaard debounce tijd wordt dus overschreven (zie int \_iDebounceTime = 100)

#### De methods

#### De properties

Alle properties van deze class zijn “Alleen lezen”. De properties kunnen dus niet door de code in een schets aangepast worden; dit kan alleen binnen de class.

##### CPreEmptiveTimer cPreEmptiveTimer

Intern heeft de CDebounce class een timer nodig

##### Int \_iPinButton;

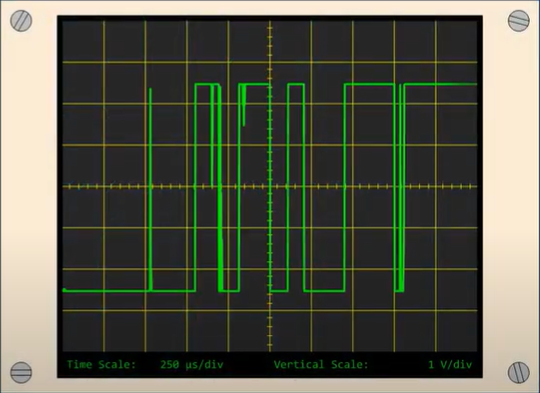
##### int \_iDebounceTime = 100

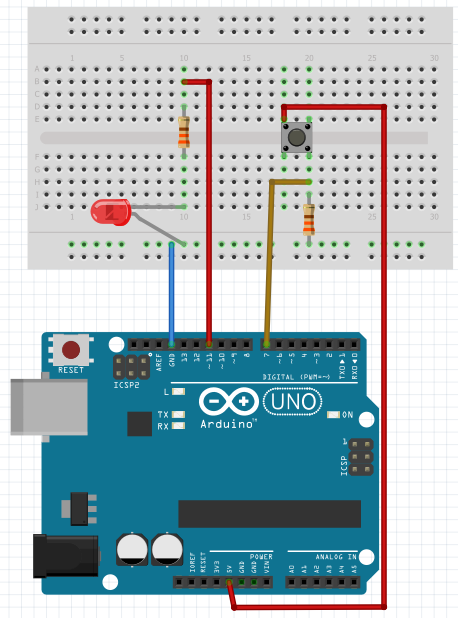
##### bool\_bState = false

# Classes

Dit hoofdstuk beschrijft enkele preEmptiveState objecten welke gebruikt kunnen worden in diverse projecten. Deze objecten maken deel uit van de meegeleverde libraries.

## CDebounce

Wanneer een aam/uit knop gebruikt wordt dan zal het contact niet altijd mooi aan of uit zijn. Door mechanische beperkingen, zoals de veerkracht van het lipje, zal de knop af en toe het contact maken of onderbreken. Dit is goed te zien op een oscilloscoop:

Wanneer de knop wordt ingedrukt zullen er momenten zijn dat de schakeling aan is maar ook soms dat de schakeling uit is. De Arduino is zo snel dat deze in staat is om dat te detecteren.

Hiervoor is de volgende Fritzing schakeling gemaakt om de debounce te ontwikkelen:

Deze staat in de file:

~\libraries\CDebounce\examples\CDebounce.fzz

De schakeling bestaat uit één button, één led en twee weerstanden. Met deze schakeling zal eerst het bouncing effect worden aangetoond.

Vervolgens zal een pre-emptive state oplossing uitgelegd worden.

### Detecteren van bouncing

Een voorbeeld programma staat in de voorbeelden – CDebounce – DebounceDetect (Zie Voorbeelden in de library)

Wanneer de knop wordt ingedrukt zou, volgens de code, de led aan gaan en bij de volgende keer dat de knop wordt ingedrukt, de led weer uitgaan. Dit gaat vaak goed echter soms, al na een paar keer indrukken zal de led-status niet veranderen (of even flikkeren); de Arduino heeft een dubbele bounce gedetecteerd.

De code om elke bounce te detecteren is vrij rechttoe – rechtaan. De hoofd functie is bounce en is geschreven op de regels 13 t/m/ 26:

void bounce(int reading) {

if(iLastReading!=reading) {

iLastReading = reading;

if(reading==HIGH) {

if(iLastBounceState==LOW) {

iLastBounceState=HIGH;

} else {

iLastBounceState=LOW;

}

digitalWrite(PIN\_LED\_BOUNCED\_UP, iLastBounceState);

}

}

}

De variabele iLastBounceState houdt bij of de led aan of uit is.

Eerst (regel 15) wordt gekeken of de status van de knop veranderd is. Dit gebeurd met de variabele iLastReading. Wanneer de knop veranderd is dan zal eerste iLastReading de nieuwe status krijgen (regel 16).

Vervolgens za, indien de knop ingedrukt is (status HIGH), één van de volgende acties moeten plaatsvinden:

* Indien de iLastBounceState HIGH is, dan zal de waarde LOW worden
* Indien de iLastBounceState LOW is, dan zal de waarde HIGH worden

Vervolgens zal de led hiermee aangestuurd worden. Dus de led veranderd van aan naar uit of van uit naar aan, elke keer dat de knop 0V of +5V is.

Maar wat blijkt nu: soms, wanneer de knop ingedrukt wordt, dan zal de led niet de goede status hebben: een bounce is dan gedetecteerd door de Arduino.

### Het oplossen van bouncing (debouncing)

Een voorbeeld programma staat in de voorbeelden – CDebounce – DebounceTestt (Zie Voorbeelden in de library)

De code is herschreven in een class. Dit wordt in XXX uitgelegd.

Een debounce statemachine moet detecteren of een knop wordt ingedrukt en of dit ingedrukt blijft. De tijd dat de statemachine moet bepalen of een knop is ingedrukt is ongeveer 100 ms. Dus wanneer er gecontroleerd wordt of een button langer dan 100 ms dezelfde status (0V / LOW / +5V of HIGH) heeft dan mag aangenomen worden dat de status van de knop niet meer veranderd.

Hiervoor kunnen we een statemachine maken. Deze ziet er als volgt uit:

Afbeelding met tekst, cirkel, diagram, lijn

Automatisch gegenereerde beschrijving

Wanneer de status 0 of 1 is, dan zal de statemachine de statuswaarde LOW hebben. In status 2 of 3 zal dit LOW zijn.

De tabel zal zijn:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | initialisatie | event | volgende status | Wat doet deze state |
| 0 | Geen initialisatie nodig | Een contact is gedetecteerd | 1 | Wachten tot een contact gedetecteerd is |
| 1 | Zet timer wit aan | Contact is uit | 0 |  |
|  |  | Timeout (100 ms) | 2 en zet status HIGH | Wacht tot contact blijvend is |
| 2 | Geen initialisatie nodig | Contact is los | 3 | Wachten tot een contact los is |
| 3 | Ze timer aan | Contact is weer aan | 2 |  |
|  |  | Timeout (100 ms) | 0 | Wacht tot contact blijvend los is |

Status 0 heeft geen initialisatie. Dat is in dit (uitzonderlijk) geval ook niet nodig:

1. De status is LOW (vanuit state 3)
2. Er is geen timeout. Tijdens elke go, wordt alleen gekeken of de waarde van de button HIGH is geworden.

De code is dan ook erg eenvoudig: (regel 62 t/m 65)

case 0:

if(bReading)

cPreEmptiveTimer.setNewState(1);

break;

Het enige dat gedaan wordt is het bepalen of de knop van LOW naar HIGH gegaan is en indien zo, dan wordt naar status 1 gesprongen.